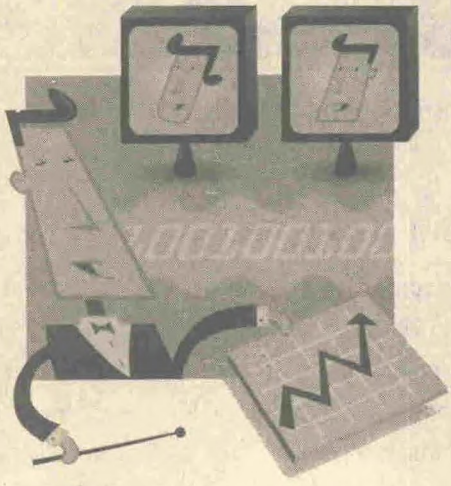
ประพนธ์ อัศวภาณุวัฒน์

SOFT MAIL

Serial Communications (ตอนที่ 1)

การใช้งาน Serial Communications Device หรือ Asynchronous Communication Device ทั้งแบบ 16 บิต และแบบ 32 บิต



สวัสดีครับ ในปัจจุบันวิวัฒนาการของอุปกรณ์เครื่องใช้ไฟฟ้าต่าง ๆ ค่อนข้างก้าวหน้ามาก จากเครื่องใช้ไฟฟ้าธรรมดา ก็เริ่มมีระบบไมโครโปรเซสเซอร์ควบคุม ทำให้สามารถประมวลผลหรือควบคุมการทำงานง่าย ๆ ได้อย่างอัตโนมัติ และมีประสิทธิภาพมาก เช่น เครื่องปรับอากาศ ซึ่งสามารถวัดอุณหภูมิและปรับคอมเพรสเซอร์ให้ผลิตความเย็น เพื่อให้ได้บรรยากาศที่เย็นสบายตามความต้องการของผู้ใช้ นอกจากนี้ยังมีอุปกรณ์เครื่องใช้ไฟฟ้าหลายชนิดที่ให้ความสำคัญกับข้อมูล โดยบางชนิดได้เพิ่มเติมเครื่องขับดิสก์ (Disk Drive) ทำให้สามารถอ่านเขียนข้อมูลลงบนแผ่นดิสก์ได้ เช่น เครื่องพิมพ์ดีดไฟฟ้ารุ่นใหม่ ที่สามารถบันทึกข้อมูลของเอกสารที่พิมพ์ ลงแผ่นดิสก์ ทำให้อ่านข้อมูลและสั่งพิมพ์ได้โดยไม่ต้องเสียเวลาพิมพ์ซ้ำอีก นอกจากนี้เครื่องใช้ไฟฟ้าบางชนิดได้เพิ่มพอร์ตสื่อสาร Serial Communication Device (RS-232) เพื่อที่สามารถติดต่อสื่อสารกับระบบคอมพิวเตอร์ได้ เช่น ตู้ชุมสายโทรศัพท์ (PABX) มีพอร์ตสื่อสารที่ใช้ส่งข้อมูลรายละเอียดการใช้โทรศัพท์ให้คอมพิวเตอร์บันทึกข้อมูล หรือสั่งพิมพ์ เป็นต้น

สำหรับพอร์ตสื่อสาร Serial Communication Device นอกเหนือจากการใช้ส่งข้อมูลระหว่างเครื่องใช้ไฟฟ้ากับคอมพิวเตอร์แล้ว ในส่วนของระบบคอมพิวเตอร์เราสามารถนำไปประยุกต์ใช้งานอื่น ๆ ได้อีกมากมาย เช่น ต่ออุปกรณ์ต่อพ่วงต่าง ๆ (เมาส์ (Mouse) ปากกาแสง (Light Pen) เครื่องอ่านบาร์โค้ด หรือแถบแม่เหล็ก (Barcode Reader/ Magnetic Reader)) และยังสามารถใช้สื่อสารระหว่างเครื่องคอมพิวเตอร์ได้

ฉบับนี้เราจะคุยกันเรื่องการใช้งาน Serial Communications Device หรือ Asynchronous Communication Device (ต่อไปนี้ผมขอเรียกสั้น ๆ ว่า "Serial Port") ตามที่ได้สัญญาไว้ในสองฉบับก่อน ทั้งแบบ 16 บิต บนระบบวินโดวส์ 3.x และแบบ 32 บิต บนระบบวินโดวส์ 95 และวินโดวส์เอ็นที โดยในตอนที่ 1 นี้จะเป็นส่วนของการโปรแกรมบนระบบวินโดวส์ 3.x และในตอนที่ 2 จะเป็นการโปรแกรมบนระบบวินโดวส์ 95 และวินโดวส์เอ็นที (ในส่วนของการโปรแกรมบนดอส (DOS) ขออนุญาตเก็บไปฝึกวิทยายุทธก่อน อาจจะได้เห็นในรูปแบบซอฟต์แวร์ชิปครับ)

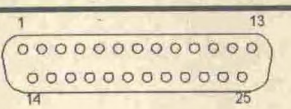
เตรียมการ

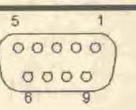
การเขียนโปรแกรมสื่อสารผ่าน Serial Port มีความต้องการพื้นฐานดังต่อไปนี้

- การทดสอบโปรแกรมที่ใช้ควบคุม Serial Port จำเป็นอย่างยิ่งที่ต้องมีพอร์ตชนิดนี้อย่างน้อย 2 พอร์ต เครื่องคอมพิวเตอร์ส่วนใหญ่จะมีพอร์ตชนิดนี้อยู่แล้ว 2 พอร์ต (ฟ้าเป็นใจ) คือพอร์ตขนาด 9 ขาสัญญาณ (9 Pins) และพอร์ตขนาด 25 ขาสัญญาณ (25 Pins) แต่สำหรับเครื่องที่ไม่มีพอร์ต PS/2 Mouse เราจำเป็นต้อง

ตารางที่ 1

แก้ไข/เพิ่มเติม ในไฟล์ SYSTEM.INI	หรือ แก้ไขเป็น
Device=serial.386	device=serial.386
Com1AutoAssign=2	COM1Irq=4
Com2AutoAssign=2	COM1Base=03F8
	COM2Irq=3
	COM2Base=02F8






```
01,10,12,13,14,15,16,17,19,21,23,24 - N/C
02 - transmit data
03 - receive data
04 - RTS (request to send)
05 - CTS (clear to send)
06 - DSR (data set ready)
07 - signal ground
08 - CD (carrier detect)
09 - +transmit current loop return
11 - -transmit current loop data
18 - +receive current loop data
20 - DTR (data terminal ready)
22 - RI (righ indicator)
25 - -receive current loop return
```

```
01 - CD in (data carrier detect)
02 - RD in (serial receive data)
03 - TD out (serial transmit
data)
04 - DTR out (data terminal
ready)
05 - gnd  (signal ground)
06 - DSR in (data set ready)
07 - RTS out (request to send)
08 - CTS in (clear to send)
09 - RI in (ring indicator)
```

ตารางที่ 2 ความหมายของขาสัญญาณของ DB 25 และ DB 9

ใช้ Serial Port นี้หนึ่งพอร์ตต่อกับเมาส์เพื่อใช้ควบคุมเคอร์เซอร์บนระบบวินโดวส์ (ฟ้ากลั่นแกล้ง) ในกรณีนี้ต้องหาเครื่องคอมพิวเตอร์เพิ่มอีกเครื่องหรือยกเลิกการใช้เมาส์ (จะยุ่งยากพอควรเมื่อใช้ระบบวินโดวส์)

- เครื่องที่ใช้ทดสอบต้องมีระบบวินโดวส์ติดตั้ง ใช้ทดสอบได้ทั้ง Windows 3.x, Windows 95 และ Windows NT (ระบบ Windows 3.x สามารถใช้ทดสอบโปรแกรมแบบ 16 บิตเท่านั้น)
- โปรแกรม Delphi เวอร์ชัน 1.0 สำหรับทดสอบโปรแกรมแบบ 16 บิต และ Delphi เวอร์ชัน 2 ขึ้นไป สำหรับทดสอบโปรแกรมในระบบ 32 บิต
- สายสัญญาณ สามารถใช้สาย Null Modem ได้ หรือสร้างขึ้นเองตามหัวข้อที่จะกล่าวถึงต่อไป

วิธีตรวจสอบว่าเมาส์เราใช้พอร์ตอะไร ให้ดูที่ปลายสายสัญญาณว่าเป็นชนิดกลม (PS/2 Mouse) หรือสี่เหลี่ยมคางหมูมี 9 รู (Serial Port 9 Pins)

ติดตั้ง Serial Port ให้กับ Windows 3.x

โดยปกติระบบ Windows จะทำการติดตั้งดีไวซ์ไดรเวอร์ (Device Driver) สำหรับควบคุม Serial Port ในขั้นตอนติดตั้งโปรแกรม Windows ให้อยู่แล้ว สำหรับ Windows 3.x ในบางกรณีอาจมีปัญหาเนื่องจากติดตั้งโปรแกรมต่าง ๆ ทำให้ข้อมูลในส่วนควบคุม Serial Port สูญหายไป เราสามารถแก้ไขเพิ่มเติมข้อมูลในส่วนนี้ได้ โดย

ระบบ Windows 3.x ตรวจสอบไฟล์ SYSTEM.INI ให้แก้ไขในส่วน [386Enh] ดังตารางที่ 1

DB 9 & DB 25 Serial Connector

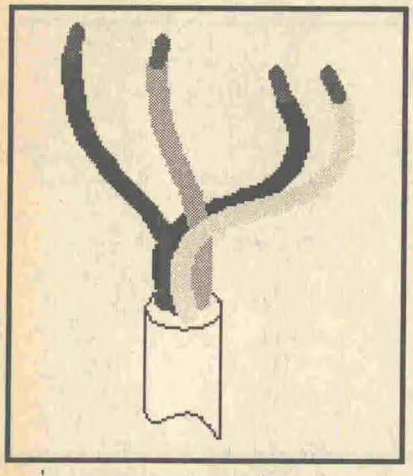
หลังจากติดตั้ง Serial Port ให้ระบบ Windows รู้จักแล้ว เราลองมาทำความรู้จักกับตัวฮาร์ดแวร์จริง ๆ บ้าง ตัว Serial Port ที่ว่านี้ฝังตัวเองอยู่ด้านหลังเครื่องคอมพิวเตอร์ มีรูปร่างลักษณะเหมือนสี่เหลี่ยมคางหมู มีเข็มยื่นออกมาจากฐานจำนวน 9 เข็ม (Pins) เราเรียกหัวชนิดนี้ว่า "หัวคอนเน็กเตอร์ ดีบีเก้าตัวผู้" (DB 9 Connector Male-Type) และชนิดที่มีเข็มจำนวน 25 เข็มยื่นออกมา เราเรียกว่า "หัวคอนเน็กเตอร์ ดีบียี่สิบห้า" (DB 25 Connector Male-Type) ต้องสังเกตดี ๆ นะครับสำหรับหัวคอนเน็กเตอร์แบบ 25 ขา เพราะมีหัวอีกประเภทมีลักษณะเหมือนกันแต่มี 25 รู (Female-Type) เป็นหัวสำหรับต่อกับเครื่องพิมพ์ชนิด Parallel Port สำหรับระบบคอมพิวเตอร์ที่ไม่มีช่องต่อเมาส์แบบ PS/2 Mouse Port เราจะใช้พอร์ตใดพอร์ตหนึ่ง (DB9 หรือ DB25) ต่อเชื่อมกับสายสัญญาณเมาส์

การต่อสายสัญญาณ

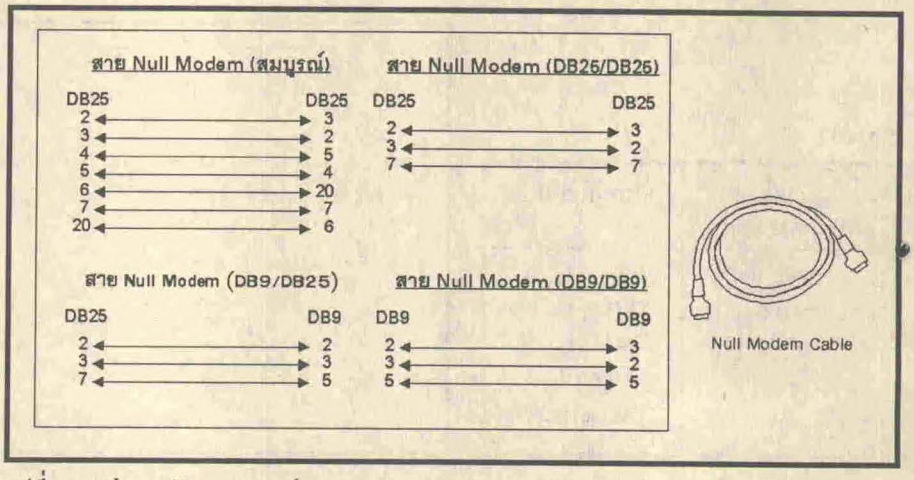
การส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ให้ได้ผลลัพธ์ที่สมบูรณ์จะต้องใช้สาย Null Modem ที่มีฉนวน (Shield) หุ้มสายสัญญาณ (Wire) ที่อยู่ด้านใน เพื่อป้องกันสัญญาณรบกวน ในกรณีที่ไม่สามารถหาสาย Null Modem เราสามารถสร้างสายสำหรับทดสอบส่งข้อมูลระยะใกล้ ๆ โดยใช้สายโทรศัพท์ (สายกลม) ที่มีสายสัญญาณ 4 เส้น และหัวคอนเน็กเตอร์ DB9/DB25 ชนิดตัวเมีย (Female-Type) จำนวน 2 ตัว แล้วต่อสายใช้ขาสัญญาณดังรูปที่ 2

การจัดหาอุปกรณ์สำหรับต่อสายสัญญาณขอให้คำนึงถึงลักษณะการใช้งาน (DB25 ↔ DB25, DB9 ↔ DB9 หรือ DB9 ↔ DB25) สำหรับสายสัญญาณที่ทำขึ้นเองด้วยสายที่ไม่มีฉนวน (เช่นสายโทรศัพท์) ไม่ควรยาวเกิน 5 เมตร อุปกรณ์ต่าง ๆ สามารถหาซื้อได้ที่บ้านหม้อ ราคาจะถูกกว่าสถานที่อื่น ๆ

เมื่อเตรียมอุปกรณ์ฮาร์ดแวร์เรียบร้อยแล้ว ก็ได้เวลาลงมือสร้างโปรแกรมแล้วครับ ในส่วนโปรแกรมจะแบ่งออกเป็น 2 ส่วน ส่วนแรกจะเป็นซอร์สโค้ด (Source Code) ของออบเจ็กต์ชื่อ TCOMM16 Object และส่วนที่สองเป็นส่วนของโปรแกรมใช้งานชื่อโปรแกรม TSTCOMM16 ครับ



รูปที่ 1 สายสัญญาณโทรศัพท์แบบ 4 Wire



รูปที่ 2 การต่อสายสัญญาณแบบต่าง ๆ

ลิสติงที่ 1 โปรแกรม TCOMM16 Objects (Ucomm16.pas) ค่าคงที่ (Const)

```
00001:  (*************************************************************)
00002:  (*   TCOMM16 Objects                                         *)
00003:  (*   ( for Windows 3.x only )                                *)
00004:  (*   Save this file as : UComm16.PAS                         *)
00005:  (*************************************************************)
00006:
00007:  unit UComm16;
00008:
00009:  interface
00010:
00011:  uses WinTypes, Messages;
00012:
00013:  const
00014:
00015:    MAX_COMM_PORT    = 4; (* Max. comm. possible in system *)
00016:    DEF_COMM_PORT    = 1; (* Default comport *)
00017:    DEF_COMM_RX_BUFF = 4096; (* 4 KB input buffer *)
00018:    DEF_COMM_TX_BUFF = 4096; (* 4 KB output buffer *)
00019:
00020:    (* user define message *)
00021:    WM_COMM_ERROR    = wm_user + $1616; (* comm. error message *)
00022:
00023:  type
00024:
00025:    TCommInfo = record
00026:      commHdl  : integer;
00027:      commPort : byte;
00028:      rxBuffer : word;
00029:      txBuffer : word;
00030:      baudRate : word;
00031:      byteSize : byte;
00032:      parity   : byte;
00033:      stopBits : byte;
00034:      errorFlg : boolean;
00035:    end;
00036:
00037:  (*- 16 bits Communications Device Control Object ----------*)
00038:
00039:    TComm16 = class
00040:    private
00041:      { Private declarations }
00042:        parentHdl : HWnd;
00043:      procedure  SendNotifyErrorMessage( errorCode : integer );
00044:    public
00045:      { Public declarations }
00046:        myCommInfo : TCommInfo;
00047:      constructor Create( wndHdl : HWnd; var commInfo : TCommInfo );
00048:      destructor Destroy; override;
00049:      function   IsDataInRxQue : word;
00050:      function   IsDataInTxQue : word;
```

→

ลิสติงที่ 1 (ต่อ)

```
00051:      function   ReadData( bufferP : pointer;
00052:                   bufferSize : integer ) : integer;
00053:      function   SendData( bufferP : pointer;
00054:                   bufferSize : integer ) : integer;
00055:    end;
00056:
00057:  implementation
00058:
00059:  uses Dialogs, SysUtils, WinProcs;
00060:
00061:  (*- TComm16 Objects ----------------------*)
00062:
00063:  constructor TComm16.Create( wndHdl:HWnd; var commInfo:TcommInfo );
00064:  var comName : string; comNameP : pChar; dcb : TDcb;
00065:      ret : integer; stat : TComStat;
00066:  begin
00067:    parentHdl:=wndHdl;
00068:    (* initialized comm. info *)
00069:    with commInfo do
00070:    begin
00071:      commHdl:=0; errorFlg:=false;
00072:      if ( not commPort in [ 1..MAX_COMM_PORT ] ) then commPort:=1;
00073:      if ( rxBuffer<1 ) or ( rxBuffer>65535 ) then
00074:        rxBuffer:=DEF_COMM_RX_BUFF;
00075:      if ( txBuffer<1 ) or ( rxBuffer>65535 ) then
00076:        txBuffer:=DEF_COMM_TX_BUFF;
00077:      if ( baudRate<1 ) then baudRate:=CBR_9600;
00078:      if ( not byteSize in [ 4..8 ] ) then byteSize:=8;
00079:      if ( not parity in [ EVENPARITY,MARKPARITY,NOPARITY,
00080:        ODDPARITY ] ) then parity:=NOPARITY;
00081:      if not stopBits in [ONESTOPBIT,ONE5STOPBITS,TWOSTOPBITS] then
00082:        stopBits:=ONESTOPBIT;
00083:    end;
00084:    myCommInfo:=commInfo;
00085:    with myCommInfo do
00086:    begin
00087:      comNameP:=@comName[1]; comName:='COM'+IntToStr(commPort)+#0;
00088:      (* open comm. port *)
00089:      commHdl:=OpenComm( comNameP, rxBuffer, txBuffer );
00090:      if ( commHdl<0 ) then
00091:      begin
00092:        myCommInfo.errorFlg:=true;
00093:        ret:=GetCommError( commHdl, stat );
00094:        SendNotifyErrorMessage( ret );
00095:        ShowMessage( 'Error on open COM'+IntToStr( commPort )+
00096:          '  error code = '+IntToStr( ret ) );
00097:        Exit;
00098:      end;
00099:    end;
00100:    (* Get comm. current settings *)
00101:    GetCommState( myCommInfo.commHdl, dcb );
00102:    with dcb do
00103:    begin
00104:      baudRate:=myCommInfo.baudRate;
00105:      bytesize:=myCommInfo.byteSize;
00106:      parity:=myCommInfo.parity;
00107:      stopBits:=myCommInfo.stopBits;
00108:    end;
00109:    (* change to my settings *)
00110:    SetCommState( dcb );
00111:    (* Clear tx/rx buffer *)
00112:    FlushComm( myCommInfo.commHdl, 0 );
00113:    FlushComm( myCommInfo.CommHdl, 1 );
00114:  end;
00115:
00116:  destructor TComm16.Destroy;
00117:  begin
00118:    if ( myCommInfo.commHdl>0 ) then
00119:      CloseComm( myCommInfo.commHdl );
00120:  end;
00121:
00122:  function TComm16.IsDataInRxQue : word;
00123:  var stat : TComStat;
00124:  begin
00125:    GetCommError( myCommInfo.commHdl,stat );
```

→

ลิสติงที่ 1 (ต่อ)

```
00126:     IsDataInRxQue:=stat.cbInQue;
00127:   end;
00128:
00129:   function TComm16.IsDataInTxQue : word;
00130:   var stat : TComStat;
00131:   begin
00132:     GetCommError( myCommInfo.commHdl,stat );
00133:     IsDataInTxQue:=stat.cbOutQue;
00134:   end;
00135:
00136:   function TComm16.ReadData( bufferP : pointer;
00137:                              bufferSize : integer ) : integer;
00138:   var ret : integer; stat : TComStat;
00139:   begin
00140:     if ( myCommInfo.errorFlg ) then exit;
00141:     ret:=ReadComm( myCommInfo.commHdl, pchar( bufferP ),
00142:                    bufferSize );
00143:     if ( ret<0 ) then
00144:     begin
00145:       myCommInfo.errorFlg:=true;
00146:       ret:=GetCommError( myCommInfo.commHdl, stat );
00147:       SendNotifyErrorMessage( ret );
00148:       ShowMessage( 'Error on received data on COM'+
00149:         IntToStr( myCommInfo.commPort )+'  error code = '+
00150:         IntToStr( ret ) );
00151:     end;
00152:     ReadData:=ret;
00153:   end;
00154:
00155:   function TComm16.SendData( bufferP : pointer;
00156:                              bufferSize : integer ) : integer;
00157:   var ret : integer; stat : TComStat;
00158:   begin
00159:     if ( myCommInfo.errorFlg ) then exit;
00160:     ret:=WriteComm( myCommInfo.commHdl, pchar( bufferP ),
00161:                     bufferSize );
00162:     if ( ret<0 ) then
00163:     begin
00164:       myCommInfo.errorFlg:=true;
00165:       ret:=GetCommError( myCommInfo.commHdl, stat );
00166:       SendNotifyErrorMessage( ret );
00167:       ShowMessage( 'Error on sent data on COM'+
00168:         IntToStr( myCommInfo.commPort )+'  error code = '+
00169:         IntToStr( ret ) );
00170:     end;
00171:     SendData:=ret;
00172:   end;
00173:
00174:   procedure TComm16.SendNotifyErrorMessage( errorCode : integer );
00175:   begin
00176:     SendMessage( parentHdl, WM_COMM_ERROR, myCommInfo.commPort,
00177:       errorCode );
00178:   end;
00179:
00180:   end.
```

TCOMM16 Object

TCOMM16 เป็นออบเจ็กต์โปรแกรมสำหรับควบคุมการรับ/ส่งข้อมูลผ่าน Serial Port แสดงอยู่ในลิสติงที่ 1 มีส่วนของโปรแกรมและกิจกรรมที่สำคัญดังต่อไปนี้

ค่าคงที่ (Const)

MAX_COMM_PORT ระบุจำนวน Serial Port ที่สามารถใช้ได้สูงสุดในระบบ ในที่นี้คือ 4 Ports

DEF_COMM_RX_BUFF ขนาด Default ของ RX Device buffer สำหรับเป็นที่พักข้อมูลขารับ (Receive) ของพอร์ต

DEF_COMM_TX_BUFF ขนาด Default ของ TX Device buffer เป็นที่พักข้อมูลขาส่ง (Transmit) ของพอร์ต

แบบข้อมูล (Type)

TcommInfo เป็นแบบข้อมูลชนิดเรคอร์ด ใช้สำหรับผ่านค่ากำหนดเบื้องต้นให้กับ Tcomm16 Object มีฟิลด์ที่สำคัญดังนี้

CommHdl กำหนดให้เป็น "0" ระบบจะทำการบันทึกค่า Comm. Handle สำหรับอ้างถึง Serial port ที่ใช้งานอยู่

CommPort ระบุ Port ที่ต้องการเปิดใช้งาน

RxBuffer ระบุขนาด Buffer ที่ต้องการใช้ในส่วนการรับข้อมูล สูงสุด 64 KB

TxBuffer ระบุขนาด Buffer ที่ต้องการใช้ในส่วนการส่งข้อมูล

BaudRate ระบุความเร็ว baud rate ที่ใช้รับ/ส่งข้อมูล เช่น 9600 BPS

ByteSize ระบุขนาดของข้อมูล (4-8 บิต)

Parity ระบุชนิดการตรวจสอบ Parity

StopBits ระบุจำนวน StopBits

กิจกรรม (Method)

constructor Create(wndHdl:HWnd; var commInfo: TcommInfo); เป็นกิจกรรมสำหรับเปิด Serial Port เพื่อใช้งานในระบบ กิจกรรมจะทำการตรวจสอบค่าพารามิเตอร์ที่ส่งให้ และทำการเปิดพอร์ตสื่อสารด้วยคำสั่ง

```
commHdl:=OpenComm( comNameP, rxBuffer, txBuffer );
```

openComm เป็นคำสั่งสำหรับเปิดการใช้งาน Serial Port ที่ต้องการระบุไว้ใน comNameP โดยกำหนดให้มีขนาด input buffer เท่ากับ rxBuffer และ output buffer เท่ากับ txBuffer ถ้าระบบดำเนินการสำเร็จจะ return ค่าพารามิเตอร์เป็น Handle ของพอร์ตที่เปิดให้กับ commHdl

destructor Destroy; กิจกรรมสำหรับยกเลิกการใช้งาน Serial Port

function IsDataInRxQue : word; กิจกรรมสำหรับตรวจสอบจำนวนข้อมูลที่ค้างอยู่ Rx Queue ด้วยคำสั่ง GetCommError ผ่านพารามิเตอร์ที่มีแบบข้อมูลเป็น TcomStat ให้กับคำสั่ง แล้วอ่านค่าจากสถานะที่ return กลับมา

function ReadData(bufferP : pointer; bufferSize : integer) : integer; กิจกรรมสำหรับอ่านข้อมูลจาก Serial Port โดยระบุตำแหน่ง bufferP ที่ใช้รับข้อมูลและขนาดของ bufferP ด้วย bufferSize (มีหน่วยเป็นไบต์) กิจกรรมจะ return จำนวนข้อมูลที่อ่านได้จริง ด้วยคำสั่ง

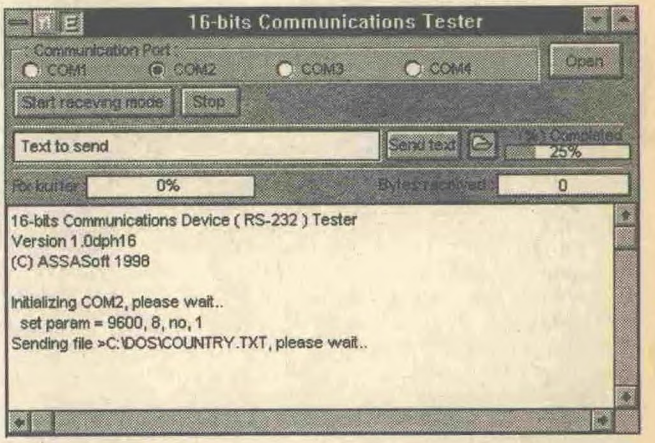
```
ret:=ReadComm( myCommInfo.commHdl,
pchar( bufferP ), bufferSize );
```

คำสั่ง ReadComm จะอ่านข้อมูลจาก commHdl ไปไว้ที่ bufferP ที่มีขนาด bufferSize แล้วส่งจำนวนข้อมูลที่อ่านได้จริงกลับให้ ret

function SendData(bufferP : pointer; bufferSize : integer) : integer; กิจกรรมสำหรับส่งข้อมูลที่อยู่ใน bufferP ความยาวเท่ากับ bufferSize ไป Serial Port โดย return จำนวนข้อมูลที่ส่งจริงกลับมาให้ด้วยคำสั่ง

```
ret:=WriteComm( myCommInfo.commHdl,
pchar( bufferP ), bufferSize );
```

คำสั่ง WriteComm จะส่งข้อมูลใน bufferP ที่มีขนาด bufferSize ให้กับ Serial Port ที่เปิดด้วย commHdl แล้ว return จำนวนที่ส่งได้จริงกลับให้ ret



รูปที่ 3

โปรแกรม TSTCOMM16

โปรแกรม TSTCOMM16 ดังรูปที่ 3 เป็นโปรแกรมที่สร้างขึ้นเพื่อทดสอบการสื่อสารทั้งการรับและการส่งข้อมูลทาง Serial Port ผ่านสายสัญญาณ Null Modem (หรือสายที่สร้างขึ้น) ภายในโปรแกรมจะเรียกใช้ TCOMM16 Objects เพื่อใช้ควบคุม Serial Port

ซอร์สโค้ดโปรแกรมแสดงอยู่ในลิสติงที่ 2 และมีส่วนของ Components ที่ใช้แสดงอยู่ในลิสติงที่ 3 สำหรับส่วนของโปรแกรมและกิจกรรมที่สำคัญมีดังนี้

ตัวแปร (VAR)

CommProc เป็น Object ที่สร้างขึ้นจาก Tcomm16 ใช้ควบคุม Serial Port

กิจกรรม (Method)

procedure commProcInit; เป็นกิจกรรมสำหรับกำหนดค่าเบื้องต้นให้ Serial Port ในโปรแกรมนี้กำหนดพอร์ตเป็น COM2, กำหนดความเร็วเท่ากับ CBR_9600 (9600 BPS), bytesize = 8, No parity และ One stop bit โดยผ่านค่าไปทาง commProcSetup และสร้าง CommProc เป็น Tcomm16 Object ด้วยคำสั่ง

```
commProc:=Tcomm16.Create( FtstComm.Handle,
commProcSetup );
```

procedure ReadDataFromcommProc; กิจกรรมสำหรับอ่านข้อมูลจาก Serial Prot ที่กำหนด กิจกรรมจะทำการตรวจสอบข้อมูลที่อยู่ใน buffer ก่อนด้วยคำสั่ง IsDataInRxQue ถ้ามีข้อมูลอยู่ก็จะทำการอ่านข้อมูลด้วยกิจกรรม CommProc.ReadData

procedure SendFile(fileName : string); กิจกรรมสำหรับส่งไฟล์ข้อมูลผ่านทางสายสัญญาณ Null Modem ให้กับเครื่อง Remote ปลายทาง

procedure TFtstComm.WaitForIncomingData(Sender: TObject; var Done: Boolean); กิจกรรมชนิด OnIdle Events จะเรียกใช้กิจกรรม ReadDdataFromCommProc เพื่ออ่านข้อมูลจาก Serial Port ในลักษณะ Idle

ลิสติงที่ 2
โปรแกรม UtstComm.p

```
00001:  (*****************************************************************)
00002:  (*  Program : Serial Communications Device Tester                 *)
00003:  (*    For Windows 3.x only                                        *)
00004:  (*  Save this file as UTSTCOMM.PAS                                *)
00005:  (*****************************************************************)
00006:
00007:  unit Utstcomm;
00008:
00009:  interface
00010:
00011:  uses
00012:    SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
00013:    Controls, Forms, Dialogs, StdCtrls, Buttons, ExtCtrls, Gauges,
00014:    uComm16;
00015:
00016:  type
00017:    TFtstComm = class(TForm)
00018:      RadioGroup1: TRadioGroup;
00019:      BitBtn1: TBitBtn;
00020:      Edit1: TEdit;
00021:      BitBtn2: TBitBtn;
00022:      BitBtn3: TBitBtn;
00023:      Bevel1: TBevel;
00024:      Bevel2: TBevel;
00025:      Label2: TLabel;
00026:      Gauge1: TGauge;
00027:      OpenDialog1: TOpenDialog;
00028:      Memo1: TMemo;
00029:      BitBtn5: TBitBtn;
00030:      Panel1: TPanel;
00031:      Label1: TLabel;
00032:      BitBtn4: TBitBtn;
00033:      Gauge2: TGauge;
00034:      Label3: TLabel;
00035:      Image1: TImage;
00036:      procedure FormCreate(Sender: TObject);
00037:      procedure FormCloseQuery(Sender: TObject;
00038:                               var CanClose: Boolean);
00039:      procedure BitBtn1Click(Sender: TObject);
00040:      procedure BitBtn3Click(Sender: TObject);
00041:      procedure BitBtn5Click(Sender: TObject);
00042:      procedure BitBtn2Click(Sender: TObject);
00043:      procedure BitBtn4Click(Sender: TObject);
00044:    private
00045:      { Private declarations }
00046:        commProc  : TComm16;
00047:        _content  : string;
00048:        txCounter : longint;
00049:      procedure commProcInit;
00050:
00051:      procedure InitVar;
00052:      procedure ReleaseVar;
00053:      procedure Echo( content : string );
00054:
00055:      procedure Display( buffer : string );
00056:      procedure ReadDataFromcommProc;
00057:      procedure SendFile( fileName : string );
00058:      procedure WaitForIncomingData(Sender: TObject;
00059:                                    var Done: Boolean);
00060:
00061:    public
00062:      { Public declarations }
00063:    protected
00064:    procedure wmCommNotifyError(var msg : TMessage); message WM_COMM_ERROR;
00065:    end;
00066:
```

→

ลิสติงที่ 2 (ต่อ)

```
00067:  var
00068:    FtstComm: TFtstComm;
00069:
00070:  implementation
00071:
00072:  {$R *.DFM}
00073:
00074:  (*----------------------------------*)
00075:
00076:  procedure Schedule;
00077:  var
00078:    Msg : TMsg;
00079:  begin
00080:    While PeekMessage(Msg,0,0,0,PM_REMOVE) do begin
00081:      TranslateMessage(Msg);
00082:      DispatchMessage(Msg);
00083:    end;
00084:  end;
00085:
00086:  (*----------------------------------*)
00087:
00088:  procedure TFtstComm.wmCommNotifyError( var msg : TMessage );
00089:  begin
00090:    Echo( 'Error occured on COM'+IntToStr( msg.wparam) +
00091:      ', error code = '+IntToStr( msg.lparam ) );
00092:  end;
00093:
00094:  (*----------------------------------*)
00095:
00096:  procedure TFtstComm.commProcInit;
00097:  var commProcSetup : TCommInfo;
00098:  begin
00099:    fillchar( commProcSetup, sizeof( TCommInfo ),0 );
00100:    with commProcSetup do
00101:    begin
00102:      commPort:=radioGroup1.ItemIndex+1;
00103:      rxBuffer:=8072;
00104:      txBuffer:=2048;
00105:      baudRate:=CBR_9600;
00106:      byteSize:=8;
00107:      parity:=NOPARITY;
00108:      stopBits:=ONESTOPBIT;
00109:    end;
00110:    commProc:=Tcomm16.Create( FtstComm.Handle, commProcSetup );
00111:    Gauge1.maxValue:=commProcSetup.rxBuffer;
00112:  end;
00113:
00114:  (*----------------------------------*)
00115:
00116:  procedure TFtstComm.InitVar;
00117:  begin
00118:    _content:=''; txCounter:=0; commProc:=nil;
00119:  end;
00120:
00121:  procedure TFtstComm.ReleaseVar;
00122:  begin
00123:    if ( commProc<>nil ) then commProc.Destroy;
00124:  end;
00125:
00126:  procedure TFtstComm.Echo( content : string );
00127:  begin
00128:    if ( memo1.lines.count>300 ) then memo1.lines.delete(0);
00129:    memo1.lines.add( content );
00130:  end;
00131:
00132:  (*----------------------------------*)
```

→

ลิสติงที่ 2 (ต่อ)

```
00133:
00134:  procedure TFtstComm.Display( buffer : string );
00135:  var c : integer;
00136:  begin
00137:    for c:= 1 to length( buffer ) do
00138:    begin
00139:      if ( buffer[c]=#10 ) then
00140:      begin
00141:        Echo( _content ); _content:='';
00142:      end else
00143:      begin
00144:        if ( length( _content )=255 ) then
00145:        begin
00146:          Echo( _content ); _content:='';
00147:        end;
00148:        _content:=_content+buffer[c];
00149:      end;
00150:    end;
00151:  end;
00152:
00153:  procedure TFtstComm.ReadDataFromcommProc;
00154:  var buffer : string; bufferP : pChar; ret : integer;
00155:  begin
00156:    if ( commProc.IsDataInRxQue>0 ) then
00157:    begin
00158:      bufferP:=@buffer[1];
00159:      while ( commProc.IsDataInRxQue>0 ) do
00160:      begin
00161:        ret:=commProc.ReadData( bufferP, 255 );
00162:        buffer[0]:=chr(ret);
00163:
00164:        Display( buffer );
00165:        inc( txCounter, ret );
00166:
00167:        Gauge1.Progress:=commProc.IsDataInRxQue;
00168:        Panel1.Caption:=IntToStr( txCounter );
00169:        Schedule;
00170:      end;
00171:    end;
00172:  end;
00173:
00174:  procedure TFtstComm.SendFile( fileName : string );
00175:  type T1KB = array[ 0..1023 ] of byte;
00176:  var fp : file; buffer : T1KB; bufferP : pointer;
00177:      readCnt : word; totalCnt : longint;
00178:  begin
00179:    if ( not fileExists( fileName ) ) then exit;
00180:    Echo( 'Sending file >'+fileName+', please wait..' );
00181:    assignFile( fp, fileName ); reset( fp,1 );
00182:    Gauge2.MaxValue:=fileSize( fp );
00183:    bufferP:=@buffer; totalCnt:=0;
00184:    repeat
00185:      blockRead( fp, buffer, sizeof( T1KB ), readCnt );
00186:      if ( readCnt>0 ) then
00187:      begin
00188:        if ( commProc.IsDataInTxQue+readCnt>2048 ) then
00189:          while ( commProc.IsDataInTxQue+readCnt>2048 ) do Schedule;
00190:        commProc.SendData( bufferP, readCnt );
00191:      end;
00192:      inc( totalCnt, readCnt ); Gauge2.Progress:=totalCnt;
00193:      Schedule;
00194:    until ( readCnt=0 );
00195:    closeFile( fp );
00196:    Echo( 'Successfully.' ); Gauge2.Progress:=0;
00197:  end;
00198:
```

→

ลิสติงที่ 2 (ต่อ)

```
00199:  procedure TFtstComm.WaitForIncomingData(Sender: TObject;
00200:                           var Done: Boolean);
00201:  begin
00202:    ReadDataFromcommProc;
00203:    Done:=false;
00204:  end;
00205:
00206:  (*-----------------------------------*)
00207:
00208:  procedure TFtstComm.FormCreate(Sender: TObject);
00209:  begin
00210:    with FtstComm do begin left:=0; top:=0; end;
00211:    InitVar;
00212:  end;
00213:
00214:  procedure TFtstComm.FormCloseQuery(Sender: TObject;
00215:                                     var CanClose: Boolean);
00216:  begin
00217:    ReleaseVar;
00218:    CanClose:=true;
00219:  end;
00220:
00221:  procedure TFtstComm.BitBtn1Click(Sender: TObject);
00222:  begin
00223:    Echo( 'Initializing COM2, please wait..' );
00224:    Echo( '  set param = 9600, 8, no, 1' );
00225:    commProcInit;
00226:  end;
00227:
00228:  procedure TFtstComm.BitBtn3Click(Sender: TObject);
00229:  begin
00230:    if ( commProc=nil ) then exit;
00231:    Application.OnIdle:=WaitForIncomingData;
00232:    Echo( 'Waiting..' );
00233:  end;
00234:
00235:  procedure TFtstComm.BitBtn5Click(Sender: TObject);
00236:  begin
00237:    if ( commProc=nil ) then exit;
00238:    Application.OnIdle:=nil;
00239:    Echo( 'Stop.' );
00240:  end;
00241:
00242:  procedure TFtstComm.BitBtn2Click(Sender: TObject);
00243:  var buffer : string; bufferP : pChar;
00244:  begin
00245:    if ( commProc=nil ) then exit;
00246:    buffer:=Edit1.text; bufferP:=@buffer[1];
00247:    commProc.SendData( bufferP, length( buffer ) );
00248:  end;
00249:
00250:  procedure TFtstComm.BitBtn4Click(Sender: TObject);
00251:  begin
00252:    if ( commProc=nil ) then exit;
00253:    if ( OpenDialog1.Execute ) then
00254:    begin
00255:      SendFile( OpenDialog1.FileName );
00256:    end;
00257:  end;
00258:
00259:  end.
```

■

ลิสติงที่ 3 รายการ Components ของฟอร์มหลักโปรแกรม UtstComm.pas ตัวแปร (VAR)

```
object FtstComm: TFtstComm
  Left = 210
  Top = 103
  BorderStyle = bsSingle
  Caption = ' 16-bits Communications Tester'
  ClientHeight = 273
  ClientWidth = 431
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clBlack
  Font.Height = -11
  Font.Name = 'Arial'
  Font.Style = []
  OnCloseQuery = FormCloseQuery
  OnCreate = FormCreate
  PixelsPerInch = 96
  TextHeight = 14
  object Label3: TLabel
    Left = 350
    Top = 62
    Width = 71
    Height = 14
    Caption = '(%) Completed'
    Transparent = True
  end
  object Bevel1: TBevel
    Left = 5
    Top = 60
    Width = 416
    Height = 6
    Shape = bsTopLine
  end
  object Bevel2: TBevel
    Left = 5
    Top = 90
    Width = 416
    Height = 6
    Shape = bsTopLine
  end
  object Label2: TLabel
    Left = 5
    Top = 95
    Width = 55
    Height = 14
    Caption = 'Rx buffer : '
    Transparent = True
  end
  object Gauge1: TGauge
    Left = 55
    Top = 95
    Width = 116
    Height = 16
    ForeColor = clRed
    Progress = 0
  end
object Label1: TLabel
    Left = 255
    Top = 95
    Width = 82
    Height = 14
    Caption = 'Bytes received : '
    Transparent = True
  end
  object Gauge2: TGauge
    Left = 340
    Top = 75
    Width = 86
    Height = 11
    ForeColor = clLime
    Progress = 0
  end
  object RadioGroup1: TRadioGroup
    Left = 3
    Top = 2
    Width = 361
    Height = 31
    Caption = ' : Communication Port : '
    Columns = 4
    ItemIndex = 1
    Items.Strings = (
      'COM1'
      'COM2'
      'COM3'
      'COM4')
    TabOrder = 0
  end
  object BitBtn1: TBitBtn
    Left = 370
    Top = 5
    Width = 51
    Height = 26
    Caption = 'Open'
    TabOrder = 1
    OnClick = BitBtn1Click
  end
  object Edit1: TEdit
    Left = 5
    Top = 65
    Width = 251
    Height = 22
    TabOrder = 2
    Text = ' Text to send'
  end
  object BitBtn2: TBitBtn
    Left = 260
    Top = 65
    Width = 51
    Height = 21
    Caption = 'Send text'
    TabOrder = 3
    OnClick = BitBtn2Click
  end
object BitBtn3: TBitBtn
    Left = 5
    Top = 35
    Width = 111
    Height = 21
    Caption = 'Start receving mode'
    TabOrder = 4
    OnClick = BitBtn3Click
  end
  object Memo1: TMemo
    Left = 0
    Top = 115
    Width = 431
    Height = 158
    Align = alBottom
    Lines.Strings = (
    '16-bits Communications Device (RS-232) Tester'
      'Version 1.0dph16'
      '(C) ASSASoft 1998'
      '')
    ScrollBars = ssBoth
    TabOrder = 5
  end
  object BitBtn5: TBitBtn
    Left = 120
    Top = 35
    Width = 41
    Height = 21
    Caption = 'Stop'
    TabOrder = 6
    OnClick = BitBtn5Click
  end
  object Panel1: TPanel
    Left = 335
    Top = 95
```

→

ลิสติงที่ 3 (ต่อ)

```
      Width = 91
      Height = 16
      BorderStyle = bsSingle
      Caption = '0'
      Color = clWhite
      TabOrder = 7
    end
    object BitBtn4: TBitBtn
      Left = 315
      Top = 65
      Width = 21
      Height = 21
      TabOrder = 8
      OnClick = BitBtn4Click
  Glyph.Data = {
        76010000424D760100000000000076000000280000002000000010000000100
        0400000000000001000012B0000120B0000100000000000000000000000000000
        8000008000000080800080000000800080008080000007F7F7F00BFBFBF000000
        FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00555555555555
        5555555555555555555555555555555555555555555555555555555555555555
        5555555555555555555555555555555555555555FFFFFFFFFF555550000000000
        5555557777777775F55500B8B8B8B8B05555775F555555575F550F0B8B8B8B8
        B05557F75F555555575F50BF0B8B8B8B8B0557F575FFFFFFFF7F50FBF0000000
        000557F557777777775550BFBFBFBFB0555557F555555557F55550FBFBFBFBF0
        555557F555555FF7555550BFBFBF00055555575F555577755555550BFBF05555
        55555575FFF755555555557000075555555555577775555555555555555555555
        5555555555555555555555555555555555555555555555555555555555555555}
      NumGlyphs = 2
    end
    object OpenDialog1: TOpenDialog
      DefaultExt = 'TXT'
      Filter = 'Text files|*.TXT|Any files|*.*'
      Left = 10
      Top = 120
    end
  end
```

วิธีการใช้โปรแกรม

หลังจากสร้างโปรแกรมเรียบร้อยแล้ว สามารถทดสอบการทำงานของโปรแกรมได้โดย

● ติดตั้งสายสัญญาณให้เรียบร้อย การต่อสายสัญญาณสามารถทำได้ 2 ลักษณะคือ

- เครื่องคอมพิวเตอร์เครื่องเดียว : ในกรณีนี้เครื่องคอมพิวเตอร์ที่ใช้จะต้องมีพอร์ตสื่อสารทั้ง 2 พอร์ต (COM1, COM2) ว่างอยู่ สามารถทำได้โดยนำหัวคอนเน็กเตอร์ด้านหนึ่งต่อกับ COM1 และอีกด้านต่อกับ COM2

- เครื่องคอมพิวเตอร์สองเครื่อง : เป็นกรณีที่เครื่องคอมพิวเตอร์มี Serial Port เหลืออยู่ช่องเดียว ให้นำหัวคอนเน็กเตอร์ด้านหนึ่งต่อกับพอร์ตที่เหลือของเครื่องแรก และอีกด้านต่อกับพอร์ตที่เหลือของเครื่องที่สอง

● รันโปรแกรม 2 ชุด โดยตัวแรกจะทำหน้าที่เป็นตัวรับ และตัวที่สองจะทำหน้าที่เป็นตัวส่งข้อมูล

● เลือก Communication Port ที่ต้องการ

● กดปุ่ม [Open]

● กดปุ่ม "Start receiving Mode" เพื่อรอรับข้อมูลโดยอัตโนมัติบนเครื่องที่ใช้รับข้อมูล (กดปุ่ม [STOP] เมื่อต้องการหยุด)

● พิมพ์ข้อความ แล้วกดปุ่ม [Send Text] บนเครื่องที่ใช้ส่งข้อมูลเมื่อต้องการส่งข้อมูลไปให้ Remote

● หรือคลิกที่ปุ่ม [แฟ้มข้อมูล] แล้วเลือกไฟล์ที่ต้องการส่งข้อมูล ระบบจะทำการส่งข้อมูลอัตโนมัติ

ถ้าโปรแกรมไม่สามารถทำงานได้ คงต้องตรวจสอบการต่อสายสัญญาณให้ถูกต้องอีกครั้ง สำหรับการใช้งาน Serial Port บนระบบปฏิบัติการ 32 บิต ผมขออนุญาตเก็บไว้คุยกันฉบับหน้า เนื่องจากชุดคำสั่งที่ใช้แตกต่างจากระบบ 16 บิตมาก ฉบับนี้คงต้องจากกันเพียงเท่านี้

สวัสดีครับ